References

- Lorenzo Seidenari
  - seidenari@dsi.unifi.it
  - http://www.micc.unifi.it/seidenari

- Giuseppe Lisanti
  - lisanti@dsi.unifi.it
  - http://micc.unifi.it/lisanti
This seminar

- Basics of Image Processing
- OpenCV basics
- Demos and samples

Needed:
- A (not so) rough knowledge of C/C++ (actually, more C than C++)
Image Processing

“Any form of signal processing for which the input is an image” [Wikipedia]

• An image is a signal
• Image processing => signal processing
• The result of image processing may or may not be an image.
• Often, the goal of image processing is feature extraction
Extracting features...

- **Feature** is a characteristic or a property of the image pixel values.

- **Feature extraction** is the process of measuring this property value over the image (or part of it).

- The goal is to exploit the feature value to extract information from the image.

- Example: is there any red object in the image? (color histogram)
Computer Vision

“Computer vision (CV) is the science and technology of machines that see” [Wikipedia]

- Computer Vision and Image Processing are NOT the same thing
- Image Processing is a tool for Computer Vision
- Computer Vision deals with image understanding: an effortless task for humans, a challenging task for computers
The challenge in CV...

<table>
<thead>
<tr>
<th>194</th>
<th>210</th>
<th>201</th>
<th>212</th>
<th>199</th>
<th>213</th>
<th>215</th>
<th>195</th>
<th>178</th>
<th>158</th>
<th>182</th>
<th>209</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>189</td>
<td>190</td>
<td>221</td>
<td>209</td>
<td>205</td>
<td>191</td>
<td>167</td>
<td>147</td>
<td>115</td>
<td>129</td>
<td>163</td>
</tr>
<tr>
<td>114</td>
<td>126</td>
<td>140</td>
<td>188</td>
<td>176</td>
<td>165</td>
<td>152</td>
<td>140</td>
<td>170</td>
<td>106</td>
<td>78</td>
<td>88</td>
</tr>
<tr>
<td>87</td>
<td>103</td>
<td>115</td>
<td>154</td>
<td>143</td>
<td>142</td>
<td>149</td>
<td>153</td>
<td>173</td>
<td>101</td>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>102</td>
<td>112</td>
<td>106</td>
<td>131</td>
<td>122</td>
<td>138</td>
<td>152</td>
<td>147</td>
<td>128</td>
<td>84</td>
<td>58</td>
<td>66</td>
</tr>
<tr>
<td>94</td>
<td>95</td>
<td>79</td>
<td>104</td>
<td>105</td>
<td>124</td>
<td>129</td>
<td>113</td>
<td>107</td>
<td>87</td>
<td>69</td>
<td>67</td>
</tr>
<tr>
<td>68</td>
<td>71</td>
<td>69</td>
<td>98</td>
<td>89</td>
<td>92</td>
<td>98</td>
<td>95</td>
<td>89</td>
<td>88</td>
<td>76</td>
<td>67</td>
</tr>
<tr>
<td>41</td>
<td>56</td>
<td>68</td>
<td>99</td>
<td>63</td>
<td>45</td>
<td>60</td>
<td>82</td>
<td>58</td>
<td>76</td>
<td>74</td>
<td>65</td>
</tr>
<tr>
<td>20</td>
<td>41</td>
<td>69</td>
<td>75</td>
<td>56</td>
<td>41</td>
<td>51</td>
<td>73</td>
<td>55</td>
<td>70</td>
<td>63</td>
<td>44</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>57</td>
<td>69</td>
<td>75</td>
<td>75</td>
<td>73</td>
<td>74</td>
<td>53</td>
<td>68</td>
<td>59</td>
<td>37</td>
</tr>
<tr>
<td>72</td>
<td>59</td>
<td>53</td>
<td>66</td>
<td>84</td>
<td>92</td>
<td>84</td>
<td>74</td>
<td>57</td>
<td>72</td>
<td>63</td>
<td>42</td>
</tr>
<tr>
<td>67</td>
<td>61</td>
<td>58</td>
<td>65</td>
<td>75</td>
<td>78</td>
<td>76</td>
<td>73</td>
<td>59</td>
<td>75</td>
<td>69</td>
<td>50</td>
</tr>
</tbody>
</table>
Why it is so difficult?

- Often there is NO SOLUTION to the vision problem: exact 3D reconstruction is not achievable from a single 2D view
- The image signal is affected by noise in several ways:
  - Lens distortions
  - Lighting conditions
  - Sensor thermal noise
  - Motion blur
  - Compression artifacts
  - ...
Visual media and compression

- Still images and video sequences are everywhere
- Gigs and gigs of informations (highly redundant!): compression is needed!
- Still images => space compression
- Video sequences => time (and space) compression
- Can Image Processing deal with compression or with compressed data?
Image Processing tool chain

Uncompress -> Filter -> Filter -> Filter -> Compress
Image Processing algorithms: a classification

- **Working in space**
  - Linear transformations (rotate, translate, scale)
  - Morphologic filters (convolution filters)
  - Histograms and statistics
  - Color transformation (contrast, brightness...)
  - Object detection
  - Many, many others...

- **Working in time**
  - Motion detection
  - Background subtraction
  - Optical flow
  - Visual tracking
  - Many, many others...
  - ...plus all the algorithms that work in space!
The programmer point of view

- An image is a 3D matrix whose elements are real values.
- The matrix dimensions are the image width, height and number of channels (1 to 4).
- The type (int, float) of the matrix values denotes the image depth (8 bit, 32 bit).
- An image is not only raw data: header information!
OpenCV Overview:

- General Image Processing Functions
- Image Pyramids
- Geometric descriptors
- Segmentation
- Camera calibration, Stereo, 3D
- Features
- Tracking
- Utilities and Data Structures
- Fitting
- Machine Learning:
  - Detection
  - Recognition
- Matrix Math

> 500 functions

opencv.willowgarage.com
The Project History

“CVL” project was started; the main goals/features:
- Human-Computer interface is a main target
- Real-Time Computer Vision Library for using by UI Developers, Videoconferences, Games
- Highly optimized for Intel Arch.

First Open Source Release: OpenCV alpha 3 at CVPR’00

OpenCV beta 1 with Linux support: CVPR’01

OpenCV 1.0 with MacOSX support

Continuous development and various research projects
OpenCV Community

The library is actively used by a large number of companies (such as Intel, IBM, Microsoft, SONY, Siemens, Google,...) and research centers (Stanford, MIT, CMU, Cambridge, INRIA etc.)

>14000 members of the forum OpenCV@yahoogroups.com, with average daily traffic ~10-20 messages.

Community contributes to the project: bug reports, patches, new features (video acquisition, 3d tracking, textures, Python interface)
# Supported Platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>IA32 (x86)</th>
<th>EM64T (x64)</th>
<th>IA64 (Itanium)</th>
<th>Other (PPC, Sparc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>✓ (w. IPP; VS98, VS2005(OpenMP), ICC, GCC, BCC)</td>
<td>✓ (w. IPP; VS98+PSDK, VS2005(OpenMP))</td>
<td>Partial Support</td>
<td>N/A</td>
</tr>
<tr>
<td>Linux</td>
<td>✓ (w. IPP; GCC, ICC)</td>
<td>✓ (w. IPP; GCC, ICC)</td>
<td>✓ (GCC, ICC)</td>
<td>✗</td>
</tr>
<tr>
<td>MacOSX</td>
<td>✓ (w. IPP, GCC, native APIs)</td>
<td>? (not tested)</td>
<td>N/A</td>
<td>✓ (iMac G5, GCC, native APIs)</td>
</tr>
<tr>
<td>Others (BSD, Solaris...)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>Reported to build on UltraSparc, Solaris</td>
</tr>
</tbody>
</table>
Example: Gesture Recognition

How to find hand?
- By color
- Using motion detection
- Depth information (w. stereo camera)

How to extract the shape?

What are the lighting conditions?

Dynamic or static gestures?

Pre-defined set of gestures
- or extendable by user?

So for this algorithm we need stereo,
3D clustering, color histograms,
statistical classification
(or just Mahalanobis distance)
and the way to visualize all the
algorithm stages for debugging

no free libraries could do all/most of this! => We had to do it ourselves

One of algorithms:
OpenCV summary

- OpenCV = Open Source Computer Vision library
- 500+ functions implementing computer vision, image processing and general-purpose numeric algorithms
- Portable and efficient (C/C++)
- Free for academic and commercial use
OpenCV refs

- OpenCV wiki
  http://opencv.willowgarage.com/wiki

- Library
  http://sourceforge.net/projects/opencvlibrary/

- Users groups
  http://tech.groups.yahoo.com/group/OpenCV/

- Book:
  Gary Bradski and Adrian Kaehler, “Learning OpenCV”, O'Reilly

- The whole world wide web!
Obtain OpenCV

• Get the stable release (currently 2.0) at http://sourceforge.net/projects/opencvlibrary/

• Install OpenCV

  • **Windows**: binary installer; samples to be compiled via CMake to get VisualStudio projects.
  
  • **Linux/OSX**: use ./configure + make + make install, or the CMake utility

  • Many common Linux distributions provide binary packages (deb and rpm), but mind the version number!
Build the samples

• **Windows:**
  
  - cl /I<opencv_inc> sample.cpp /link /libpath:<opencv_lib_path> cxcore.lib cv.lib highgui.lib
  
  - Or, create a VisualStudio project (remember to set the paths for include files and libraries)

• **Linux/OSX:**

  - g++ -o sample `pkg-config --cflags --libs opencv` sample.cpp

• **Both:**
  
  - Use a good IDE (Eclipse works just fine!)
OpenCV design

CV
Image processing and Vision Algorithms

MLL
Statistical Classifiers and Clustering Tools

HighGUI
GUI, Image and Video I/O

CXCORE
basic structures and algorithms, XML support, drawing functions
OpenCV basic structures

- Multichannel and multidimensional matrices (common base type: CvArr):
  - CvMat, CvMatND, CvSparseMat
  - IplImage

- Geometrical entities:
  - CvPoint, CvPoint2D32f, CvPoint3D32f, CvPoint2D64f, CvPoint3D64f
  - CvRect
  - CvSize, CvSize2D32f
  - CvScalar
IplImage

- nSize = sizeof(IplImage)
- nChannels
- depth
- alphaChannel
- colorModel
- channelSeq
- dataOrder
- origin
- align
- width
- height
- roi
- imageSize
- imageData
- widthStep
- BorderMode
- BorderConst
- imageDataOrigin
IplImage

- **nSize** = sizeof(IplImage)
- **nChannels**: number of channels. Most OpenCV functions support 1-4 channels.
- **alphaChannel**: ignored by OpenCV
- **depth**: pixel depth in bits (IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16U, IPL_DEPTH_16S, IPL_DEPTH_32S, IPL_DEPTH_32F, IPL_DEPTH_64F)
- **colorModel**: ignored by OpenCV
- **channelSeq**: ignored by OpenCV
- **dataOrder**: 0 – interleaved color channels; 1 – separate color channels
- **origin**: 0 – top left origin; 1 – bottom right origin (Windows style)
- **align**: ignored by OpenCV
IplImage

- **width**: image width in pixels
- **height**: image height in pixels
- **roi**: Region Of Interest
- **imageSize**: image data size in bytes (height*widthStep)
- **imageData**: pointer to aligned image data
- **widthStep**: the size of an aligned image row, in bytes
- **BorderMode**: border completion mode, ignored by OpenCV
- **BorderConst**: border completion mode, ignored by OpenCV
- **imageDataOrigin**: a pointer to the origin of the image data (not necessarily aligned)
IplImage: rows alignment

- Image raw data is stored in a linear address space: matrices are vectorized.
- If the image width is not a multiple of the memory word size, dummy bytes are added so to align image rows with memory words.
- widthStep > width
OpenCV color spaces

http://www.poynton.com/notes/colour_and_gamma/ColorFAQ.html

- RGB <=> CIE XYZ
- RGB <=> YCrCb (YCC)
- RGB <=> HSV
- RGB <=> HLS
- RGB <=> CIE L*a*b*
- RGB <=> CIE L*u*v*
- RGB <=> Bayer
An example: jet color map

Jet color map is commonly used to represent gray level images by mapping pixels intensity into a color tone.
Affine transformation

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} = \text{mapMatrix} \cdot \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

if CV_WARP_INVERSE_MAP is not set

\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix} = \text{mapMatrix} \cdot \begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix}
\]

otherwise

- mapMatrix is 2x3 floating point matrix
- Use cv2DRotationMatrix to get mapMatrix
- Use cvWarpAffine to apply the transformation
OpenCV 2.x C++ Interface

- IplImage is replaced by cv::Mat
- cv::Mat:
  - Easier memory management (RAII)
  - Shallow copied with assignement/copy constructor
  - Operator overloading
    - Mat A;
    - VideoCapture capture();
    - capture >> A; A=A+100; //increase brightness
- STL containers
  - Mat A; vector<Mat> planes; split(A,planes);

- Still fully compatible with old interface
- Some functionalities are only available with this interface.